DTU Library

A route-based decomposition for the Multi-Commodity k-splittable Maximum Flow Problem

Gamst, Mette

Published in:
Proceedings

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

A route-based decomposition for the Multi-Commodity k-splittable Maximum Flow Problem.

M. Gamst

Technical University of Denmark,
DTU Management Engineering
gamst@man.dtu.dk

Abstract. The Multi-Commodity k-splittable Maximum Flow Problem routes flow through a capacitated graph such that each commodity uses at most k paths and such that the total amount of routed flow is maximized. This paper proposes a branch-and-price algorithm based on a route-based Dantzig-Wolfe decomposition, where a route consists of up to k paths. Computational results show that the new algorithm has best performance on seven benchmark instances and is capable of solving two previously unsolved instances.

Keywords: *Branch and price; Dantzig-Wolfe decomposition; Multi-commodity flow; k-splittable; Combinatorial optimization*

1 Introduction

The $\mathcal{NP}$-hard Multi-Commodity k-splittable Maximum Flow Problem (denoted MCkMFP) maximizes the amount of routed flow in a capacitated graph using at most k paths per commodity. Given is a graph $G = (V, E)$, where each edge $(ij) \in E$ has capacity $u_{ij} > 0$. Given is also a set of commodities L, each consisting of a source s_l and a target vertex t_l. The MCkMFP has application in *Multi-Protocol Label Switching* in telecommunications [9]. The problem was introduced by Baier et al. [1]. Much work has sinze been performed on non-exact solution methods for (variants of) the problem, see e.g. [1,2,3,7,8]. Exact approaches in the literature consist of path-based branch-and-price algorithms, see [4,5,9]. The algorithms all have bottlenecks in the form of symmetry in the solution space or complicated branching and large branch-and-bound trees.

In this paper a new Dantzig-Wolfe decomposition for the MCkMFP is proposed. The goal of the decomposition is to reduce the branching difficulties of the branch-and-price algorithms in the literature. The method is inspired by a similar approach for the the Routing and Wavelength Assignment Problem [6].

2 Solution approach

The decomposition in this paper is based on routes, where each column in the master problem is a route consisting of up to k paths with given flow for a given commodity. The set of columns for each commodity l is denoted R^l. Variable $x_r^l \in \{0,1\}$ indicates if route r for commodity l is part of a solution. Positive constants δ_r^{ij} and δ_r denote the amount of flow traveling on edge (ij) and the total amount of flow carried by route r, respectively. The master problem is:

$$\max \quad \sum_{l \in L} \sum_{r \in R^l} \delta_r x_r^l \tag{1}$$

$$\text{s.t.} \sum_{l \in L} \sum_{r \in R^l} \delta_r^{ij} x_r^l \leq u_{ij} \ \forall (ij) \in E \tag{2}$$

$$\sum_{r \in R^l} x_r^l \leq 1 \qquad \forall l \in L \tag{3}$$

$$x_r \in \{0,1\} \qquad \forall r \in R \tag{4}$$

(1) maximizes the amount of routed flow, (2) ensures that edge capacities are satisfied, (3) ensures at most one route per commodity, and (4) forces all variables to take on feasible values. An incumbent solution is found using a greedy heuristic: for each commoditity the shortest path problem is solved on a reduced graph, which iteratively includes edges with lower and lower capacity. Edge capacities are updated with each found path.

Branching is applied in case of fractionality. This occurs, if a commodity l is not fully routed and then cuts are added: $\sum_{r \in R^l} x_r^l = 0$ vs. $\sum_{r \in R^l} x_r^l = 1$. Fractionality also occurs, if more than one route is used for a commodity l. This is remedied by forbidding or forcing edge usage through cuts: $\sum_{r \in R^l:(ij) \in r} x_r^l = 0$ vs. $\sum_{r \in R^l:(ij) \in r} x_r^l = 1$. If branching is still necessary, we use the path-based strategy presented in [5] forbidding or forcing usage of paths $b \in P$.

The dual variables of constraints (2) and (3) are denoted $\pi_{ij} \geq 0$ resp. $\lambda_l \geq 0$. The duals of the branching cuts are $\beta_l \in \mathbb{R}$, $\beta_{ij} \in \mathbb{R}$ and $\beta_b \in \mathbb{R}$, respectively. The pricing problem must decide the amount of routed flow: let $f_{ij} \geq 0$ denote the amount of flow on edge (ij). The reduced cost for commodity l is:

$$\sum_{(s_l j) \in E} f_{s_l j} - \sum_{(ij) \in E} (\pi_{ij} f_{ij} - \beta_{ij} w_{ij}) - \lambda_l - \beta_b w_b \geq \lambda_l + \beta_l \tag{5}$$

where $w_{ij} \in \{0,1\}$ denotes whether or not edge (ij) is used by any path in the generated route column, and $w_b \in \{0,1\}$ indicates if path b is part of the generated route column. The pricing problem consists of generating and assigning flow to up to k paths wrt. the reduced costs (5). This corresponds to the $\mathcal{NP}$-hard single-commodity k-splittable maximum flow problem. The pricing problem is solved heuristically using an algorithm similar to the incumbent but with edge weights modified to match the reduced cost. When the heuristic fails, the pricing problem is solved to optimality by applying CPLEX on a mathematical formulation. The formulation consists of the edge-based formulation of the single commodity k-splittable flow problem [9] with the reduced cost (5) as objective function and with extra constraints to detect usage of paths $b \in P$.

3 Computational evaluation

The computational evaluation is performed on benchmark instances from the literature [9]. Results are seen in Table 1. The new algorithm is denoted RBP and is compared to the leading algorithms from the literature: (3BP) by [9], (2BP) by [4] and (2BCP) by [5]. The table content is: benchmark name, value of k, optimal solution value, size and depth of branch-and-bound tree, and time usage in seconds. An "-" indicate no solution due to excessive time or memory issues and **bold** results indicate best performance.

The new algorithm has best performance on seven instances and it is capable of solving two previously unsolved instances, i.e., the single-commodity instances `Random20-140` for $k = 6$ and $k = 7$. As can be seen from the results, the new branch-and-price algorithm does not branch much, which is a significant improvement compared to the results from the literature. The algorithm is competitive with the leading algorithms from the literature on smaller instances, but it does not scale well. Solving the pricing to optimality using CPLEX is the reason for the large time usage.

References

1. G. Baier, E. Kohler, and M. Skutella. On the k-splittable flow problem. *Algorithmica*, 42:231–248, 2005.
2. M. Caramia and A. Sgalambro. A fast heuristic algorithm for the maximum concurrent k-splittable flow problem. *Optimization Letters*, 4:37–55, 2010.
3. C. Duhamel and P. Mahey. Multicommodity flow problems with a bounded number of paths: A flow deviation approach. *Networks*, 49(1):80–89, 2007.
4. M. Gamst, P. N. Jensen, D. Pisinger, and C. Plum. Two- and three-index formulations of the minimum cost multicommodity k-splittable flow problem. *European Journal of Operations Research*, 202(1):82–89, 2010.
5. M. Gamst and B. Petersen. Comparing branch-and-price algorithms for the multicommodity k-splittable maximum flow problem. *European Journal of Operational Research*, 217(2):278–286, 2012.
6. B. Jaumard, C. Meyer, and B. Thiongane. On column generation formulations for the RWA problem. *Discrete Applied Mathematics*, 157:1291–1308, 2009.
7. R. Koch, M. Skutella, and I. Spenke. Maximum k-splittable s, t -flows. *Theory of Computing Systems*, 43(1):55–66, 2008.
8. F. Salazar and M. Skutella. Single-source k-splittable min-cost flows. *Operations research letters*, 37:71–74, 2009.
9. J. Truffot and C. Duhamel. A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optimization*, 5(3):629–646, 2008.

Table 1. Results from solving the single-commodity `Random, tg` instances, and the multi-commodity `Random instances`, respectively

		3BP			2BP			2BCP			RBP		
Problem	**z***	**size**	**depth**	**time**	**size**	**depth**	**time**	**size**	**depth**	**time**	**size**	**depth**	**time**
Random10-45													
k =1	73	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.01
k =2	142	5	2	**0.01**	4	1	**0.01**	8	2	**0.01**	1	0	0.07
k =3	209	9	3	**0.02**	21	3	0.03	20	3	**0.02**	1	0	0.22
k =4	260	45	17	0.08	411	12	0.56	34	4	**0.03**	1	0	0.41
k =5	306	369	22	0.80	23599	18	44.96	40	4	**0.07**	1	0	0.58
k =6	345	973	26	2.90	>427099	>26	-	135	6	**0.22**	1	0	1.23
k =7	381	4281	36	16.55	>354551	>22	-	313	8	**0.64**	1	0	8.64
k =8	413	22985	43	102.51	>431299	>29	-	606	9	**1.31**	1	0	171.83
k =9	429	>110199	>58	-	>388228	>26	-	2507	11	**5.97**	1	0	-
k =10	451	>104999	>57	-	>456699	>41	-	2355	12	**5.91**	1	0	-
Random15-60													
k =1	86	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.01
k =2	163	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.03
k =3	221	9	3	**0.02**	41	6	0.06	12	2	**0.02**	1	0	0.22
k =4	248	111	10	0.32	>100454	>26	-	111	6	**0.22**	1	0	1.25
k =5	268	557	18	551.83	>176599	>29	-	322	7	**0.76**	1	0	4.03
k =6	287	419	21	1.59	>277801	>31	-	354	9	**0.79**	1	0	26.35
k =7	295	19097	35	72.91	>387565	>23	-	836	10	**1.74**	1	0	820.34
k =8	301	>88799	>47	-	>413343	>33	-	4995	11	**11.32**	1	0	-
k =9	306	>153099	>51	-	>547079	>28	-	2263	11	**4.42**	1	0	-
Random20-140													
k =1	81	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.01
k =2	158	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.02
k =3	228	1	0	0.02	1	0	**0.00**	1	0	**0.00**	1	0	0.04
k =4	253	9935	31	75.25	>41444	>42	-	90	18	**1.04**	1	0	1.23
k =5	274	>39999	>41	-	>68299	>66	-	819	22	12.65	1	0	**12.32**
k =6	294	>30199	>61	-	>60299	>86	-	>14106	>32	-	1	0	**28.99**
k =7	311	>28999	>70	-	>75894	>46	-	>14299	>32	-	1	0	**264.32**
k =8	319	>30599	>80	-	>94699	>101	-	4028	22	**52.95**	1	0	-
k =9	325	>39599	>93	-	>108990	>63	-	130	9	**0.32**	1	0	-
k =10	327	2907	109	19.15	>272685	>49	-	17	3	**0.02**	1	0	9.69
k =11	327	1325	86	8.75	49	3	**0.03**	20	5	**0.03**	1	0	0.70
Problem	**z***	**size**	**depth**	**time**	**size**	**depth**	**time**	**size**	**depth**	**time**	**size**	**depth**	**time**
tg40-1													
k =1	517	1	0	**0.00**	1	0	0.01	1	0	**0.00**	1	0	0.32
k =2	750	5	2	0.07	4	1	**0.02**	10	3	0.07	1	0	87.10
k =3	908	>9999	>40	-	>83282	>61	-	231	11	**3.32**	1	0	-
k =4	994	>7799	>57	-	>82770	>45	-	893	18	**25.15**	1	0	-
k =5	1004	15	7	0.09	703	27	1.41	11	2	**0.03**	1	0	0.10
k =6	1004	1	0	0.03	29	3	**0.02**	43	6	0.07	1	0	0.13
tg40-5													
k =1	487	1	0	0.01	1	0	**0.00**	1	0	**0.00**	1	0	0.35
k =2	828	>20599	>46	-	>64248	>45	-	144	9	1.49	1	0	**0.43**
k =3	1062	>17299	>59	-	>77103	>44	-	276	8	4.20	1	0	**2.60**
k =4	1078	181	47	0.61	>148934	>22	-	1520	21	26.53	1	0	**0.08**
k =5	1078	3	1	**0.03**	61	4	0.04	76	20	1.72	1	0	0.10
tg80-1													
k =1	549	1	0	0.04	1	0	**0.02**	1	0	**0.02**	1	0	621.54
k =2	984	1591	29	65.22	2308	22	59.16	288	11	**8.72**	1	0	-
k =3	1411	>2199	>36	-	>51476	>49	-	1914	10	**110.38**	1	0	-
Problem	**z***	**size**	**depth**	**time**	**size**	**depth**	**time**	**size**	**depth**	**time**	**size**	**depth**	**time**
Random11-42													
k =1	291	5	2	0.02	7	3	**0.01**	7	2	0.02	5	2	0.07
k =2	343	1	0	**0.01**	7	2	**0.01**	6	1	**0.01**	1	0	0.03
k =3	344	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.05
k =4	344	1	0	**0.00**	1	0	**0.00**	1	0	**0.00**	1	0	0.06
Random20-80													
k =1	347	7	3	0.06	3	1	**0.02**	7	2	0.04	3	1	0.30
k =2	553	3	1	0.03	4	1	0.02	4	1	**0.01**	5	2	1.18
k =3	584	1063	24	6.14	57	7	**0.16**	1020	16	3.45	3	1	1.66
k =4	601	5599	33	40.05	1041	10	2.02	>81550	>548	-	1	0	**1.83**
k =5	617	13291	44	117.96	4363	14	**7.35**	49695	34	198.61	23	10	110.87
k =6	621	>48999	>40	-	3998	11	**6.42**	32552	29	100.08	7	3	297.34
k =7	626	413	37	3.48	17	2	**0.02**	116	14	0.22	1	0	-
k =8	626	1	0	0.03	1	0	**0.01**	1	0	**0.01**	1	0	2.19
Random22-56													
k =1	365	9	3	**0.02**	7	3	**0.02**	7	2	**0.02**	5	2	0.18
k =2	389	11	4	0.02	10	3	0.02	9	3	**0.01**	17	4	1.09
k =3	407	1	0	**0.01**	1	0	**0.01**	1	0	**0.01**	3	1	0.35
k =4	407	1	0	0.01	1	0	**0.00**	1	0	**0.00**	3	1	0.43